IN THE

UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICANT:	William M. Cullen and David A. Chappell
SERIAL NO.:	09/993,865
FILING DATE:	November 14, 2001
TITLE:	Message Handling
EXAMINER:	Dhairya A. Patel
GROUP ART UNIT:	2151
ATTY. DKT. NO.:	23982-10313

CERTIFICATE OF EFS-WEB TRANSMISSION			
Pursuant to 240 OG 45 and the *Legal Framework For EFS-Web*, I hereby certify that this follow-on correspondence is being officially submitted through the USPTO EFS-Web system from the Pacific Time Zone of the United States on the local date shown below.			
Signature:	**/Greg T. Sueoka/**		
Typed or Printed Name:	**Greg T. Sueoka, Reg. 33,800**	Dated:	**September 3, 2009**

MAIL STOP APPEAL BRIEF-PATENTS
COMMISSIONER FOR PATENTS
P.O. BOX 1450
ALEXANDRIA, VA 22313-1450

APPEAL BRIEF

Pursuant to the requirements of 37 C.F.R. § 41.37, please consider this document as the Appellants' Brief in the present application currently before the Board of Patent Appeals and Interferences (hereinafter "the Board").

Real Party in Interest

The real party in interest in this Appeal is Progress Software Corporation, a corporation of Massachusetts.

Related Appeals and Interferences

There are no related appeals, interferences or judicial proceedings known to Appellant, Appellant's legal representative or the Assignee that may directly affect, be directly affected by or have a bearing on the Board's decision in the pending appeal.

Status of Claims

Claims 1 and 3-31 are pending in this Application and stand rejected. On April 29, 2009, the appellants appealed from the final rejection of claims 1 and 3-31. These claims are set forth in an appendix attached hereto.

Status of Amendments

No amendments have been filed subsequent to final rejection mailed on February 3, 2009.

Summary of Claimed Subject Matter

The following explains the subject matter of each independent claim with references to supporting aspects of the specification.

Claim 1 recites a method of handling a message received at a messaging system server (4:23-25), the method comprising: storing, in non-persistent storage, the message (2:10-15, 5:5-23, Fig. 2-4); determining whether the message has been delivered (2:10-15, 5:15-23); if the message has been delivered, removing the message from the non-persistent storage (5:5-23, 7:4-15, Fig. 5); and after a configurable delay interval has elapsed and if the message has not been removed from the non-persistent storage, saving the message to persistent storage so that the message can be retrieved and delivered (5:5-23, 9:7-14, Fig. 8).

Claim 13 recites a method of handling guaranteed messages received at a message-oriented middleware server over a network (4:23-25), the method comprising: storing, in a log queue in non-persistent storage, guaranteed messages received from at least one client as the guaranteed messages are received (2:10-15, 5:5-23, Fig. 2-4); determining whether one of the guaranteed messages has been delivered (2:10-15, 5:15-23); if the guaranteed message has been delivered, removing the message from the non-persistent storage (5:5-23, 7:4-15, Fig. 5); dynamically determining a delay time period (9:7-14); after the determined delay period has elapsed and if the message has not been removed from the non-persistent storage, saving the guaranteed message to persistent storage so that the guaranteed message can be retrieved and delivered (5:5-23, 9:7-14, Fig. 8); and transmitting a guarantee acknowledgement message to a client that sent the received guaranteed message, the guarantee acknowledgement message indicating that the received guaranteed message will not be lost by the server (7:20-25).

Claim 16 recites a computer program product, disposed on a computer readable medium, for handling messages received at a server (4:23-25, 12:10-13:5), the computer program including instructions for causing a server processor to: store, in a non-persistent storage, messages received from at least one client as the messages are received (2:10-15, 5:5-23, Fig. 2-4); determine whether one of the messages has been delivered (2:10-15, 5:15-23); if the message has been delivered, remove the message from the non-persistent storage (5:5-23, 7:4-15, Fig. 5); and after a configurable delay period has elapsed and if the message has not been removed from the non-persistent storage, save the message to persistent storage so that the message can be retrieved and delivered (5:5-23, 9:7-14, Fig. 8).

Claim 24 recites a message oriented middleware server, the server comprising: non-persistent storage; persistent storage; at least one processor; and instructions for causing the server processor to (4:23-25, 12:10-13:5): store, in the non-persistent storage, messages received from at least one client as the messages are received (2:10-15, 5:5-23, Fig. 2-4); determine whether one of the messages has been delivered (2:10-15, 5:15-23); if the message has been delivered, remove the message from the non-persistent storage (5:5-23, 7:4-15, Fig. 5); and after a configurable delay period has elapsed and if the message has not been removed from the non-persistent storage, save the message to persistent storage so that the message can be retrieved and delivered (5:5-23, 9:7-14, Fig. 8).

Grounds of Rejection to be Reviewed on Appeal

Claims 1, 3, 7-12, 16-17, 20-25 and 28-31 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Chandrasekaran in view of Hamada, further in view of Kalkunte.

Argument

103(a) Rejection Based on Chandrasekaran, Hamada and Kalkunte

The Examiner rejected claims 1, 3, 7-12, 16-17, 20-25 and 28-31 under 35 U.S.C. § 103(a) as being unpatentable over Chandrasekaran in view of Hamada, further in view of Kalkunte.

Claim 1 recites:

A method of handling a message received at a messaging system server, the method comprising:
storing, in non-persistent storage, the message;
determining whether the message has been delivered;
if the message has been delivered, removing the message from the non-persistent storage; and

after a configurable delay interval has elapsed and if the message has not been removed from the non-persistent storage, saving the message to persistent storage so that the message can be retrieved and delivered.

A message is stored in non-persistent storage. A determination is made regarding whether the message has been delivered. If the message has been delivered, the message is removed from the non-persistent storage. If the message has not been delivered after a configurable delay time period, the message is saved to persistent storage.

Initially storing a message in non-persistent storage enables rapid access to the message for retransmission during the delay interval. This enables message retransmission without introducing additional latency caused by storing the message to and retrieving the message from persistent storage. Since initially storing the message in non-persistent storage reduces the time needed to access the message, overall system performance is increased. However, the non-persistent storage is limited and therefore the message queue cannot grow indefinitely in non-persistent storage. Accordingly, the delivered messages are removed from the non-persistent storage and undelivered messages that have not been delivered within a delay interval are moved to persistent storage. Saving the undelivered messages to persistent storage after the delay interval has elapsed beneficially frees space in the non-persistent storage for recent messages, which increases system performance while also providing continued access to older messages from the persistent storage for later retrieval and delivery. Additionally, storing the message to persistent storage beneficially ensures that the message will not be lost in the event of network, system or power failure and makes the message accessible for subsequent analysis or delivery attempts.

Chandrasekaran does not disclose, teach or suggest "if the message has been delivered, removing the message from the non-persistent storage" because determining a site failure does not disclose confirming the message's delivery.

Chandrasekaran discusses a system for propagating (transmitting) a message from a source site 200 to a destination site 202 while efficiently maintaining and using metadata associated with the message (abstract). The source site tracks messages using a propagation queue 204 and a propagation table 212 (7:61-63; FIG. 2A). The propagation queue, which is in volatile memory, stores the message that is awaiting transmission (7:63-65). The propagation table, which is in non-volatile memory, does not store the message itself. The propagation table instead stores the metadata including "a sequence number attribute 228, a UID attribute 230 and a state attribute 232." (FIG. 2A, 9:5-9:7). The sequence number is based on the time at which the message was inserted in propagation queue (7:2-7:26). This use of time based sequence number avoids the inefficient use of sequence numbers based on atomic counters and locks. *See* 4:13-4:643 (impossible to generate a sequence number for the message atomically; using exclusive locks and shared locks still not enough to solve this problem).

In Chandrasekaran, at the source site: First, the message is "dequeued" (removed) from the propagation queue (7:17-19). Next, the message is assigned a propagation sequence number (7:19-21). The message is then transmitted to destination site (7:28-30). The propagation sequence number, the UID, and an initial propagation state are then stored in the non volatile memory in propagation table at the source site (7:30-32). The source site then sends a commit request to the destination site (Fig. 3, 11:23-11:25). Upon receiving a commit reply from the destination site, the source site updates the propagation state information in its non-volatile memory (11:33-11:37).

At the destination site, the received message is "enqueued" (added) in receive message queue (12:4-12:5). The destination site then waits for the commit request from the source site. If the destination site does not get a commit request and the destination site determines that the source site has failed, the destination site dequeues the message from the receive message queue (12:24-12:31). Otherwise, upon receiving the commit request from the source site, "the received message data is stored into non-volatile memory at destination site." (12:33-12:35). The destination site then sends a commit reply to indicate that the changes contained in the received message have been committed (12:47-12:50).

In the Final Office Action dated 2/3/09, the Examiner cites Chandrasekaran at 12:24-12:31 as disclosing "if the message has been delivered, removing the message from the non-persistent storage." Again, Chandrasekaran at 12:24-12:31 discloses removing the message from receive message queue if the source site has failed and not if the message has been delivered. Accordingly, the requirement in Chandrasekaran for removing the message is completely different from the requirement for removing the message in claimed invention. Chandrasekaran therefore does not disclose, teach, or suggest "**if the message has been delivered**, removing the message from the non-persistent storage."

It would not be obvious from Chandrasekaran to infer the claimed element because removing the messages from volatile memory because of failure of source site does not disclose removing the message from volatile memory if a message has been delivered. The destination site in Chandrasekaran does not deliver the message to any other entity. It would not make any sense for destination site to track the delivery of a message as the destination site does not deliver the message to any entity. Accordingly, one of ordinary skill in the art would not modify the destination site to deliver the received message to another entity, maintain a copy of the delivered message in volatile memory, track the delivery of the message and remove the message from volatile memory if the message is not delivered.

Chandrasekaran does not disclose, teach or suggest "after a configurable delay interval has elapsed and if the message has not been removed from the non-persistent storage, saving the message to persistent storage so that the message can be retrieved and delivered."

The Examiner cites Chandrasekaran at 7:30-39 as disclosing the above mentioned limitation. Chandrasekaran does not disclose the above mentioned limitation because saving "propagated message data" is not saving "message data." Chandrasekaran clearly states that the "propagated message data" is not the same as "message." Chandrasekaran defines "message" as a "combination of the message data and its associated header information." (7:11-7:13). Chandrasekaran then defines "propagated message data" as "propagation sequence number, the UID, and an initial propagation state." (7:30-7:34). Because Chandrasekaran stores the propagated message data (i.e. the metadata) and not the message itself to non-volatile memory, Chandrasekaran does not disclose, teach or suggest "saving the message to persistent storage **so that the message can be retrieved and delivered**." The examiner is reading the emphasized language out of the claim.

Assuming *arguendo* that the Examiner is justified in interpreting "propagated message data" in Chandrasekaran as "message" in the invention, Chandrasekaran still does not disclose the above mentioned limitation because the propagated message data is not stored for later delivery. Chandrasekaran instead stores propagated message data in non-volatile memory because the storage "allows the source site to determine, even after a source site failure, whether a particular message has previously been propagated to the destination site." (7:34-7:38). The source site does not store the propagated message data so that the source site can retrieve and deliver the propagated message data to the destination site. This is because the source site uses propagated message data for internal housekeeping and not for delivering the propagated message data to external entities like destination site. The source site therefore does not "retrieve[] and deliver[]" propagated message data. In other words, the source site does not store the propagated message data "to persistent storage so that the [propagated message data] can be retrieved and delivered." Moreover the propagated message data does not include the message payload and the propagated message data alone cannot be used to construct and deliver the message. Accordingly, Chandrasekaran does not disclose, teach, or suggest "saving the message to persistent storage so that the message can be retrieved and delivered."

Additionally, Chandrasekaran cannot be modified to save the message in non-volatile memory because such a modification changes the principal operation of Chandrasekaran. MPEP 2143.01. The principal operation of Chandrasekaran is to reduce inefficiencies involved with creating and maintaining appropriate accurate metadata. *See* Chandrasekaran, 4:13-4:643 (impossible to generate a sequence number for the message atomically; using exclusive locks and shared locks still not enough to solve this problem). To avoid these inefficiencies, Chandrasekaran maintains selective metadata in non-volatile memory and updates the metadata at appropriate times. Saving the message to non-volatile memory in addition to the metadata introduces additional inefficiencies involved with maintaining locks on the message. Because saving message to non-volatile memory introduces additional inefficiencies instead of reducing them, saving the message to non-volatile memory impermissibly changes the principal operation of Chandrasekaran.

Hamada does not disclose and Hamada cannot be impermissibly modified to disclose "if the message has not been removed from the non-persistent storage, saving the message to persistent storage so that the message can be retrieved and delivered"

Hamada discusses a "redundant message processing system." (abstract). In the Hamada system, both the transmitter and the receiver side have a current system and a stand-by system (Fig 21, 17:15-17:21). Both the current system and the stand-by system share a common non-volatile memory (Fig. 19, 15:50-15:52, Fig. 21, 17:25-17:28). The current system stores the content of the transmitted or received messages to this common non-volatile memory (17:25-17:28). Now, if the current transmitter system goes down, the stand-by transmitter system reads the transmitted message from the common non-volatile memory and retransmits the message to the receiver (Fig. 22, 17:66-18:12). In this manner, the Hamada system delivers a redundant message processing system.

To make sure that the redundant or stand-by system in Hamada is able to re-transmit the message, the Hamada system must store **every** message in non-volatile memory before transmitting the message. *See* 17:35-17:65 ("the content of message ID and the content of message are … stored in the non-volatile memory … [and] then a request is made … for a message transmission."). Accordingly, every message that is transmitted is first stored in persistent storage.

The Examiner impermissibly modifies Hamada's principal operation and argues that the impermissible modification discloses "**if the message has not been removed from the non-persistent storage, saving the message to persistent storage.**" As discussed above, Hamada does not selectively store only the undelivered messages to the persistent storage and instead stores **every** message to persistent storage. There is not selection or choice whatsoever in Hamada as to whether messages get stored to persistent storage. Hamada cannot be modified to store only some of the messages to persistent storage because storing every message in persistent memory is critical for the Hamada redundant message processing system. If Hamada's current system does not store all messages in non-volatile memory, the stand-by system would not be able to redeliver already transmitted messages because the messages may not exist in non-volatile memory. The Hamada system therefore would not be functional without storing every message in non-volatile memory. Thus, the Hamada system cannot be impermissibly modified for storing only those messages that have "not been removed from non-persistent storage … after a configurable delay interval has elapsed." Accordingly Hamada does not disclose and Hamada cannot be impermissibly modified and combined with Chandrasekaran to disclose "if the message has not been removed from the non-persistent storage, saving the message to persistent storage so that the message can be retrieved and delivered."

Kalkunte does not disclose and Kalkunte cannot be impermissibly modified to disclose "after a configurable delay interval has elapsed … saving the message to persistent storage so that the message can be retrieved and delivered."

Kalkunte discloses a method and apparatus for adding a **randomized** propagation delay for avoiding data packet collision in a network. Kalkunte, title, abstract. The servers and clients request data after random delays to avoid requesting the data at the same time and overloading the network. Kalkunte, abstract.

The Examiner cites Kalkunte at 5:15-5:44 and 6:65-6:67 as disclosing "after a **configurable delay** interval has elapsed and if the message has not been removed from the non-persistent storage, saving the message to persistent storage so that the message can be retrieved and delivered." (emphasis added). Kalkunte, however, uses the time delay for a completely different purpose. Kalkunte uses time delay to send messages from the server and clients in a cluster at varying time instances (abstract). Kalkunte does not use the time delay to move messages from volatile memory to non-volatile memory.

Additionally, the time delay in Kalkunte is randomized and not configurable (title, abstract, 6:60-7:1). The configurable time delay in the claimed invention provides an administrator the option to balance the efficiencies and risks involved with storing the messages in volatile memory for a given period of time. The longer the message is stored in volatile memory, the higher the risk that the message may be lost in case of system failure. Kalkunte is not concerned with providing any such balancing option and Kalkunte is instead concerned with avoiding network data collision. Accordingly, Kalkunte uses random and not configurable time delay to restart servers at different times. Because the time delay is random, Kalkunte does not disclose a time delay that can be configured. Kalkunte therefore does not disclose, teach, or suggest "after a **configurable delay** interval has elapsed and if the message has not been removed from the non-persistent storage, saving the message to persistent storage so that the message can be retrieved and delivered."

Moreover, one of ordinary skill in the art would not be motivated to combine Chandrasekaran, Hamada and Kalkunte to achieve a method where "after a **configurable delay** interval has elapsed and if the message has not been removed from the non-persistent storage, saving the message to persistent storage so that the message can be retrieved and delivered." The lack of motivation is apparent from Examiner's piecemeal treatment of a single claim element, the Examiner's need to cite three references for three artificially created parts of the claim element, and modification of the teachings from two of those references to argue the disclosure of the single element.

One of ordinary skill in the art would have to impermissibly modify Hamada to achieve parts of the claimed element. Additionally, without impermissible hindsight, there is no reason in Kalkunte to change the random delay interval to configurable delay interval because Kalkunte works based on restarting the servers after random time periods and not configurable time periods. If every server was configured to send messages after the same configurable time period, the configuration would defeat the purpose of avoiding traffic from the restarted servers as all the servers would send messages to each other after the same time period. The Examiner's motivation to combine Kalkunte with Chandrasekaran and Hamada is a classic example of impermissible hindsight reconstruction because it is apparent from the claim that the configurable delay interval allows the administrator to configure the amount of delay period after which the message can be stored to the persistent storage. However, this motivation cannot be derived from Kalkunte, Hamada or Chandrasekaran as none of the references address the problem of efficient message transmission by moving the messages from non-persistent to persistent memory. Accordingly, barring impermissible hindsight, there is no motivation to combine Chandrasekaran, Hamada and Kalkunte to disclose the "after a **configurable delay** interval has elapsed and if the message has not been removed from the non-persistent storage, saving the message to persistent storage so that the message can be retrieved and delivered."

In sum, Chandrasekaran, Hamada and Kalkunte, individually or in combination, do not disclose, teach or suggest "if the message has been delivered, removing the message from the non-persistent storage" or "after a configurable delay interval has elapsed and if the message has not been removed from the non-persistent storage, saving the message to persistent storage so that the message can be retrieved and delivered." The Examiner cites other references like Stein for other claims but the Examiner does not cite any other references for these elements.

Therefore, claim 1 is patentable over Chandrasekaran, Hamada, and Kalkunte alone and in combination. Independent claims 16 and 24 recite similar language and are also patentable over Chandrasekaran, Hamada, and Kalkunte alone and in combination, for at least the same reasons.

Conclusion

The arguments presented herein demonstrate that claims 1 and 3-31 of the present application are patentable over the prior art of record. Therefore, Appellants respectfully request that the Board reverse the Examiner's rejections of these claims.

Respectfully submitted,
WILLIAM M. CULLEN, ET AL

Dated: September 3, 2009 /Greg T. Sueoka/

Greg T. Sueoka, Reg. No. 33,800
Attorney for Applicant
Fenwick & West LLP
Silicon Valley Center
801 California Street
Mountain View, CA 94041
Tel.: (650) 335-7194
Fax.: (650) 938-5200

Claims Appendix

1. (Rejected) A method of handling a message received at a messaging system server, the method comprising:

storing, in non-persistent storage, the message;

determining whether the message has been delivered;

if the message has been delivered, removing the message from the non-persistent storage; and

after a configurable delay interval has elapsed and if the message has not been removed from the non-persistent storage, saving the message to persistent storage so that the message can be retrieved and delivered.

2. (Canceled)

3. (Rejected) The method of claim 1, wherein storing in the non-persistent storage comprises storing in a log queue.

4. (Rejected) The method of claim 1, further comprising transmitting an acknowledgement message to a client that sent the received message, the acknowledgement message indicating that the received message will not be lost by the server in the case of server failure.

5. (Rejected) The method of claim 4, wherein transmitting the acknowledgment message to the client comprises transmitting the acknowledgment message to the client for successful delivery of the received message.

6. (Rejected) The method of claim 4, wherein transmitting the acknowledgment message to the client comprises transmitting the acknowledgment message to the client for storage of the received message in persistent storage.

7. (Rejected) The method of claim 1, further comprising determining the delay interval.

8. (Rejected) The method of claim 7, wherein determining the delay interval comprises:

determining at least one metric based on messages handled by the server; and

determining the delay interval based on the at least one metric.

9. (Rejected) The method of claim 8, wherein the metric comprises a metric based on a number of sending clients using the server to deliver messages.

10. (Rejected) The method of claim 7, wherein determining the delay interval comprises dynamically determining the delay.

11. (Rejected) The method of claim 1, wherein the message was received over a communications network.

12. (Rejected) The method of claim 1,

wherein the message comprises a guaranteed message; and

wherein the messaging system comprises a message-oriented middleware system.

13. (Rejected) A method of handling guaranteed messages received at a message-oriented middleware server over a network, the method comprising:

storing, in a log queue in non-persistent storage, guaranteed messages received from at least one client as the guaranteed messages are received;

determining whether one of the guaranteed messages has been delivered;

if the guaranteed message has been delivered, removing the message from the non-persistent storage;

dynamically determining a delay time period;

after the determined delay period has elapsed and if the message has not been removed from the non-persistent storage, saving the guaranteed message to persistent storage so that the guaranteed message can be retrieved and delivered; and

transmitting a guarantee acknowledgement message to a client that sent the received guaranteed message, the guarantee acknowledgement message indicating that the received guaranteed message will not be lost by the server.

14. (Rejected) The method of claim 13, wherein transmitting the guarantee acknowledgement message comprises:

if the guaranteed message was successfully delivered, transmitting the guarantee acknowledgement message; and

if the guaranteed message was not successfully delivered, transmitting the guarantee acknowledgement message when the guaranteed message is persistently stored.

15. (Rejected) The method of claim 13, wherein dynamically determining the delay time period comprises:

determining a metric based on messages handled by the server; and

determining the delay time period based on the determined metric.

16. (Rejected) A computer program product, disposed on a computer readable medium, for handling messages received at a server, the computer program including instructions for causing a server processor to:

store, in a non-persistent storage, messages received from at least one client as the messages are received;

determine whether one of the messages has been delivered;

if the message has been delivered, remove the message from the non-persistent storage; and

after a configurable delay period has elapsed and if the message has not been removed from the non-persistent storage, save the message to persistent storage so that the message can be retrieved and delivered.

17. (Rejected) The computer program of claim 16, wherein the instructions for causing the server processor to store messages in the non-persistent storage comprise instructions for causing the server processor to store the messages in a log queue.

18. (Rejected) The computer program of claim 16, further comprising instructions for causing the server processor to transmit an acknowledgement message to a client that sent the received message, the acknowledgement message indicating that the received message will not be lost by the server.

19. (Rejected) The computer program of claim 18, wherein the instructions for causing the server processor to transmit the acknowledgment message to the client comprise instructions for causing the server processor to transmit the acknowledgment message to the client for a message saved from non-persistent storage to persistent storage.

20. (Rejected) The computer program of claim 16, further comprising instructions for causing the server processor to determine the delay.

21. (Rejected) The computer program of claim 20, wherein the instructions for causing the server processor to determine the delay comprise instructions for causing the server processor to:

determine at least one metric based on the received messages; and

determine the delay based on the at least one metric.

22. (Rejected) The computer program of claim 21, wherein the metric comprises a metric based on a number of clients using the server to deliver messages.

23. (Rejected) The computer program of claim 16, wherein the instructions for causing the processor to determine the delay comprise instructions for causing the processor to dynamically determine the delay.

24. (Rejected) A message oriented middleware server, the server comprising:

non-persistent storage;

persistent storage;

at least one processor; and

instructions for causing the server processor to:

store, in the non-persistent storage, messages received from at least one client as the messages are received;

determine whether one of the messages has been delivered;

if the message has been delivered, remove the message from the non-persistent storage; and after a configurable delay period has elapsed and if the message has not been removed from the non-persistent storage, save the message to persistent storage so that the message can be retrieved and delivered.

25. (Rejected) The server of claim 24, wherein the instructions for causing the server processor to store the messages in the non-persistent storage comprise instructions for causing the server processor to store the messages in a log queue.

26. (Rejected) The server of claim 24, further comprising instructions for causing the server processor to transmit an acknowledgment message to a client that sent the received message, the acknowledgment message indicating that the received message will not be lost by the server.

27. (Rejected) The server of claim 26, wherein the server instructions for causing the server processor to transmit the acknowledgment message to the client comprise instructions for causing the server processor to transmit the acknowledgment message to the client as the message is stored from nonpersistent storage to persistent storage.

28. (Rejected) The server of claim 24, further comprising instructions for causing the server processor to determine the delay.

29. (Rejected) The server of claim 28, wherein the instructions for causing the server processor to determine the delay comprise instructions for causing the server processor to:

determine at least one metric based on the received messages; and

determine the delay based on the at least one metric.

30. (Rejected) The server of claim 29, wherein the metric comprises a metric based on a number of clients using the server to deliver messages.

31. (Rejected) The server of claim 24, wherein the instructions for causing the processor to determine the delay comprise instructions for causing the processor to dynamically determine the delay.

Evidence Appendix

None

<u>**Related Proceedings Appendix**</u>

None